UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/292,951	11/09/2011	Jerome F. DULUK JR.	NVDA/SC-11-0069-US1	6201

102324 7590 02/01/2017

Artegis Law Group, LLP/NVIDIA
7710 Cherry Park Drive Suite T #104
Houston, TX 77095

EXAMINER	
ALFRED, MELISSA A	

ART UNIT	PAPER NUMBER
2199	

NOTIFICATION DATE	DELIVERY MODE
02/01/2017	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

kcruz@artegislaw.com
ALGdocketing@artegislaw.com
mmccauley@artegislaw.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Ex parte JEROME F. DULUK JR., LACKY V. SHAH,
and SEAN J. TREICHLER

Appeal 2016-002675
Application 13/292,951[1]
Technology Center 2100

Before CARLA M. KRIVAK, KEVIN C. TROCK, and
JOHN R. KENNY, *Administrative Patent Judges*.

TROCK, *Administrative Patent Judge*.

DECISION ON APPEAL

Introduction

Appellants seek review under 35 U.S.C. § 134(a) from the Examiner's Final Rejection of claims 1–20. We have jurisdiction under 35 U.S.C. § 6(b).

We AFFIRM.

---

[1] Appellants indicate the Real Party in Interest is NVIDIA Corporation. App. Br. 3.

Invention

The claimed invention relates to the execution of compute tasks, specifically, to the encapsulation of a compute task state. Spec. ¶ 1.

Exemplary Claim

Exemplary claim 1 is reproduced below with disputed limitations emphasized:

1. A method of encapsulating and scheduling compute tasks in a streaming multiprocessor, the method comprising:

allocating memory for storing a metadata structure for a compute task;

storing initialization parameters in the metadata structure that configure the streaming multiprocessor to execute the compute task;

storing scheduling parameters in the metadata structure that control the scheduling of the compute task;

storing execution parameters in the metadata structure that control execution of the compute task by the streaming multiprocessor; and

scheduling the compute task based on the scheduling parameters in the metadata structure, for execution in the steaming multiprocessor based on the execution parameters in the metadata structure.

Rejections

Claims 1, 7, 10, 11, 17, and 20 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Duluk Jr. (US 7,697,007 B1; Apr. 13, 2010).

Claims 2–4, 8, 9, 12–14, 18, and 19 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Duluk and Sonnier et al. (US 2010/0293353 A1; Nov. 18, 2010).

Claims 5, 6, 15, and 16 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Duluk, Sonnier, and Bates et al. (US 2007/0074207 A1; Mar. 29, 2007).

ANALYSIS

We have reviewed the Examiner's rejections and the evidence of record in light of Appellants' arguments that the Examiner has erred. We disagree with Appellants' arguments and conclusions. We adopt as our own (1) the findings and reasons set forth by the Examiner in the Office Action from which this appeal is taken and (2) the findings and reasons set forth in the Examiner's Answer. We concur with the conclusions reached by the Examiner and further highlight specific findings and argument for emphasis as follows.

Section 102 – Independent Claims 1, 10, 11

Appellants contend the Examiner erred in rejecting independent claims 1, 10, and 15 because Duluk fails to disclose storing initialization parameters, scheduling parameters, and execution parameters in a metadata structure. App. Br. 11–12; Reply Br. 3. Appellants argue that the Examiner maps the claimed parameters to the state information disclosed by Duluk, but that Duluk merely stores state information in a conventional buffer and is entirely silent regarding storing state information in any type of metadata structure. App. Br. 11.

The Examiner finds, however, and we agree, Duluk discloses the claimed parameters because state information, as explained by Duluk, includes any information (other than input data) relevant to defining a Cooperative Thread Array (CTA). Ans. 15. The Examiner finds Duluk

discloses that such state information includes, for example, "parameters that define the size of the CTA, the amount of register file space required for each thread, a starting program counter (e.g., memory address) for a program to be executed by each thread, and selection of hardware resource allocation algorithms." *Id.* (citing Duluk 4:33–39).

Moreover, the Examiner notes, and we agree, Appellants' Specification does not specifically define a "metadata structure." The Examiner concludes, and we also agree, a metadata structure can include a buffer under the broadest reasonable interpretation. Ans. 16 (citing Duluk 1:41–42 ("a pushbuffer can be used as a mechanism to queue the launching of Cooperative Thread Arrays"); 3:45–47 ("pushbuffer 150 operates as a first-in, first-out FIFO buffer for thread launch commands to GPU 122"). Further, Appellants provide no reason why a metadata structure cannot include a buffer. App. Br. 11; Reply Br. 3.

Therefore, we are not persuaded the Examiner erred in finding Duluk discloses storing initializing parameters, scheduling parameters, and execution parameters in a metadata structure within the meaning of claims 1, 10, and 11, and we, therefore, sustain the Examiner's rejection of these claims under 35 U.S.C. § 102(b).

Section 103 – Dependent Claims 6, 16

Appellants contend the Examiner erred in rejecting dependent claims 6 and 16 because the combination of Duluk, Sonnier, and Bates does not teach or suggest "receiving pointers including a pointer to the metadata structure and additional pointers to additional metadata structures that are associated with additional compute tasks," as recited in those claims. App. Br. 12, 13; Reply Br. 4, 5. Appellants argue the task definitions taught by Bates merely

store pointers to other memory locations that store various types of information. App. Br. 12. These task definitions, Appellants argue, do not store initialization, scheduling, or execution parameters, and therefore, cannot properly be equated to the recited metadata structure. *Id.*

The Examiner finds, however, and we agree, Bates teaches task definitions that *provide pointers to metadata structures* that include storing initialization, scheduling, and execution parameters. Ans. 18. The Examiner finds that Bates teaches:

> task definitions 118, may include pointers to memory locations containing task parameters 120 and SPU task code image 122 ... The task parameters 120 may include information related to the task, including, but not limited to input/output (110) addresses, 110 sizes, addresses for input and output task data 123 and the like. The STM kernel 112 loads code 124 into the SPU 104 using the code image 122 and parameters 120 the SPU 104 where they are stored as context data 126. The SPU 104 can then run the code 124 to load and process the task data 123; and [0044], Information in the task definition 118 directs the STM to main memory addresses corresponding to the SPU task parameters 120 and task code image 122).

Ans. 18 (citing Bates ¶¶ 38, 44).

Moreover, the Examiner does not rely on Bates to teach the recited parameters. As discussed above, the Examiner relies on Duluk to teach the recited initialization, scheduling and execution parameters. Bates' task definitions provide the pointers to the metadata structures. Ans. 18.

Accordingly, we are not persuaded the Examiner erred in finding the combination of Duluk, Sonnier, and Bates teaches or suggests "receiving pointers including a pointer to the metadata structure and additional pointers to additional metadata structures that are associated with additional compute

tasks," as recited in dependent claims 6 and 16. Therefore, we sustain the Examiner's rejection of these claims under 35 U.S.C. § 103(a).

Improper Combination

Appellants contend that modifying the system of Duluk in the manner proposed by the Examiner would both render the system inoperable for its intended purpose and fundamentally change the principle operation of Duluk. App. Br. 13, 14; Reply Br. 5, 6. Appellants argue that because Duluk stores state information and GPU commands in a single command buffer, modifying Duluk with the pointers of Bates would result in a system where each of the pointers would point to the same single command buffer in which they are stored, effectively destroying the purpose of the pointers. App. Br. 13. Because Duluk's buffer does not include distinct metadata structures, Appellants argue, Duluk cannot be modified to implement multiple pointers and one of ordinary skill in the art would never attempt to modify Duluk in the manner proposed by the Examiner. App. Br. 13, 14.

The test for obviousness, however, is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference. *In re Keller*, 642 F.2d 413, 425 (CCPA 1981) (citations omitted). "[I]t is not necessary that the inventions of the references be physically combinable to render obvious the invention under review." *In re Sneed*, 710 F.2d 1544, 1550, 218 USPQ 385, 389 (Fed. Cir. 1983). "Combining the teachings of references does not involve an ability to combine their specific structures." *In re Nievelt*, 482 F.2d 965, 179 USPQ 224, 226 (CCPA 1973). Rather, the test for obviousness is what the combined teachings of the references would have suggested to those of ordinary skill in the art. *In re Keller*, 642 F.2d 413, 425 (CCPA 1981)

(citations omitted).

Here, the Examiner finds, and we agree, Duluk teaches a method for launching CTA's based on state information loaded into a pushbuffer. Ans. 19. The Examiner also finds that Duluk teaches a pushbuffer can be implemented as a *plurality* of memory structures and that threads can indirectly access memory. Ans. 19, 20 (citing Duluk 3:37–44; 4:13–15 (emphasis added)). The Examiner also finds, and we agree, Sonnier teaches a job scheduling method where task data structures stored in memory are assigned to one of several execution queues for processing and that these task data structures include pointers. Ans. 20 (citing Sonnier ¶¶ 23, 27, 29, 79). The Examiner further finds, and we agree, Bates teaches loading a task definition from shared memory for execution on one of several processing devices using a task queue. *Id.* (citing Bates Abstract, ¶ 36). The Examiner concludes, and we agree, it would have been obvious to one of ordinary skill in the art to combine the job scheduling methods of Duluk and Sonnier with the task queues of Bates to schedule tasks that include pointers to the task data. Appellants' unsupported argument that one of ordinary skill in the art could not or would not have modified Duluk to implement multiple pointers and a plurality of memory structures is simply unpersuasive in light of the evidence adduced by the Examiner.

Remaining Claims 2–5, 7–9, 12–15, 17–20

Appellants have not presented separate arguments with respect to dependent claims 2–5, 7–9, 12–15, and 17–20. *See* App. Br. 11–14. We, therefore, are not persuaded that the Examiner erred in rejecting these claims. *See* 37 C.F.R. § 41.37(c)(1)(iv); *In re Lovin*, 652 F.3d 1349, 1356 (Fed. Cir. 2011) ("We conclude that the Board has reasonably interpreted

Rule 41.37 to require applicants to articulate more substantive arguments if they wish for individual claims to be treated separately."). Accordingly, we sustain the Examiner's rejection of these claims.

DECISION

We AFFIRM the Examiner's rejections of claims 1–20.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED